Inhibited Effects in CP-logic

Wannes Meert[1] and Joost Vennekens[2]

[1]Dept. Computer Science,
[2]Dept. Computer Science – Campus De Nayer,
KU Leuven, Belgium
{wannes.meert,joost.vennekens}@cs.kuleuven.be

Abstract. An important goal of statistical relational learning formalisms is to develop representations that are compact and expressive but also easy to read and maintain. This is can be achieved by exploiting the modularity of rule-based structures and is related to the noisy-or structure where parents independently influence a joint effect. Typically, these rules are combined in an additive manner where a new rule increases the probability of the effect. In this paper, we present a new language feature for CP-logic, where we allow negation in the head of rules to express the inhibition of an effect in a modular manner. This is a generalization of the inhibited noisy-or structure that can deal with cycles and, foremost, is non-conflicting. We introduce syntax and semantics for this feature and show how this is a natural counterpart to the standard noisy-or. Experimentally, we illustrate that in practice there is no additional cost when performing inference compared to a noisy-or structure.

Keywords: Statistical Relational Learning, Bayesian networks, Noisy-or, Inhibited noisy-or, CP-logic

1 Introduction

Statistical Relational Learning (SRL) [7] and probabilistic logic learning [2] are concerned with representations that combine the benefits of probabilistic models, such as Bayesian networks, with those of logic representations, such as first-order logic. In this work we focus on the family of SRL formalisms that associate probabilities to directed logic programming rules and can be interpreted as cause-effect pairs (e.g. CP-logic [17], ProbLog [6] or PRISM [14]).

An important goal of these formalism is to develop representations that are easy to read and maintain. One way in which they attempt to achieve this is by exploiting the inherent modularity of the rule-based structure of logic programs. We will illustrate this using CP-logic because of its intuitive, causal interpretation but the results are generally applicable. A CP-logic theory consists of a set of rules, and each rule is viewed as an independent causal mechanism. This makes it easy to update an existing theory by adding a (newly discovered) causal mechanism, since none of the existing rules have to be touched. In certain restricted cases, a theory in CP-logic can be translated into a Bayesian network in a very straightforward way. The translation may preserve this modularity property by

using *noisy-or* nodes to represent the joint effect of different rules with the same head (i.e., different rules that may independently cause the same effect). It has been shown that the use of such noisy-or nodes in Bayesian networks makes it easier for human experts to supply probabilities and build more accurate models [18]. This is further evidence for the importance of this modularity property in probabilistic logics.

Currently, however, CP-logic's modularity is limited, in the sense that each new rule that is added for a given effect can only *increase* its probability. In practice, it occurs just as often that an existing theory has to be modified because a previously unknown mechanism makes some effect *less* likely in certain cases. CP-logic currently offers no modular way of adding such a new mechanism to an existing theory—it always requires changes to existing rules. In this paper, we present a new language feature, where we allow *negation in the head* of rules. We develop a syntax and semantics for this feature, and demonstrate that it indeed extends the modularity property to the discovery of new mechanisms that *decrease* the probability of existing events. In the special case where the CP-logic theory can easily be translated to a Bayesian network, we show that this feature of negation in the head reduces to an inhibited noisy-or structure [4]. While this is a little known kind of node in the literature on probabilistic graphical models, our analysis shows that it is a natural counterpart to the standard noisy-or. Additionally, we show experimentally that this intuitive structure exhibits the same advantageous properties as a noisy-or structure such as a linear number of parameters and inference that is polynomial in the number of parents.

2 Preliminaries and Motivation

CP-logic offers a compact and robust way of specifying certain kinds of probability distributions. This is due to three interacting properties:

- Different causes for the same effect can be represented as separate rules, each with their own probabilities, which are combined with a *noisy-or* when necessary. This leads to a modular representation.
- Logical variables may be used to write down first-order rules that serve as templates for sets of propositional rules. In this way, very compact representations can be achieved.
- The semantics of CP-logic is defined in a robust way, allowing, in particular, also cycles in the possible cause-effect relations.

To illustrate these properties, consider the following example.

Example 1. An infectious disease spreads through a population as follows: whenever two people are in regular contact with each other and one is infected, there is a probability of 0.6 of the infection spreading also to the other person. Given a set of initially infected people and a graph of connections between individuals in the population, the goal is to predict the spread of the disease.

In CP-logic, this can be represented by a set of two rules:

$$(Inf(x) : 1.0) \leftarrow InitialInf(x). \tag{1}$$
$$(Inf(x) : 0.6) \leftarrow Contact(x, y) \wedge Inf(y). \tag{2}$$

Given any set of individuals and any interpretation for the exogenous predicates *InitialInf* and *Contact*, this CP-theory defines the probability with which each individual will be infected. In particular, no restrictions (such as acyclicity) are imposed on the *Contact*-relation.

In addition to representing probability distributions in a compact way, CP-logic also aims at being *elaboration tolerant*: once a CP-theory for a given domain has been constructed, it should be easy to adapt this theory when we learn new facts about the domain. Ideally, new knowledge should be incorporated in a way which respects the inherent modularity of CP-logic, in the sense that it may involve adding or removing rules, but not changing existing rules.

One such operation for which CP-logic is obviously well-suited is when a new cause for some effect is discovered. For instance, suppose we learn that, in addition to being among the initially infected and having contact with infected individuals from the population, people may also contract the disease by travelling to particular locations (e.g., with probability 0.2). We can update our CP-logic model accordingly, by simply adding an additional rule:

$$(Inf(x) : 1.0) \leftarrow InitialInf(x).$$
$$(Inf(x) : 0.6) \leftarrow Contact(x, y) \wedge Inf(y).$$
$$(Inf(x) : 0.2) \leftarrow RiskyTravel(x).$$

Importantly, there is no need to change our existing rules.

A second operation is discovering that certain parts of the population form an exception to the general rules. For instance, suppose that certain people are discovered to be especially susceptible (e.g., probability 0.8) to contracting the disease through contact with an already infected person. We can represent this by "case splitting" rule (2) into the following two rules:

$$(Inf(x) : 0.6) \leftarrow Contact(x, y) \wedge Inf(y) \wedge \neg Susceptible(x).$$
$$(Inf(x) : 0.8) \leftarrow Contact(x, y) \wedge Inf(y) \wedge Susceptible(x).$$

However, this solution has the downside that it forces us to change an existing rule. A better alternative is to exploit the additive nature of different causes for the same effect in CP-logic:

$$(Inf(x) : 0.6) \leftarrow Contact(x, y) \wedge Inf(y).$$
$$(Inf(x) : 0.5) \leftarrow Contact(x, y) \wedge Inf(y) \wedge Susceptible(x).$$

For non-susceptible individuals, only the first rule is applicable, so they still get infected with the same probability of 0.6 as before. The same rule of course also applies to susceptible individuals, whom the second rule then gives an *additional*

probability of getting infected *because* they are susceptible. This brings their total probability of being infected up to $0.6 + (1 - 0.6) \cdot 0.5 = 0.8$. When compared to the "case splitting" theory, this representation has the advantage that it allows the "default" rule for normal people to remain unchanged.

In addition to discovering that certain parts of the population are especially susceptible to the infection, it is equally possible to discover that certain people tend to be more resistant to it. Again, this can be solved by case splitting:

$$(\mathit{Inf}(x) : 0.6) \leftarrow \mathit{Contact}(x, y) \wedge \mathit{Inf}(y) \wedge \neg \mathit{Resistant}(x).$$
$$(\mathit{Inf}(x) : 0.4) \leftarrow \mathit{Contact}(x, y) \wedge \mathit{Inf}(y) \wedge \mathit{Resistant}(x).$$

A solution in which we can keep our original "default" rule unchanged is not possible using noisy-or or is not intuitive to impossible in current probabilistic logics. Indeed, this is an obvious consequence of the fact that adding additional rules can only *increase* probabilities. In this paper, we introduce the new feature of negation in the head of rules, which will allow us to represent also a *decrease* in probabilities. In particular, we will be able to represent our example as:

$$(\mathit{Inf}(x) : 0.6) \leftarrow \mathit{Contact}(x, y) \wedge \mathit{Inf}(y).$$
$$(\neg \mathit{Inf}(x) : 1/3) \leftarrow \mathit{Resistant}(x).$$

3 Preliminaries: Formal Semantics of CP-logic

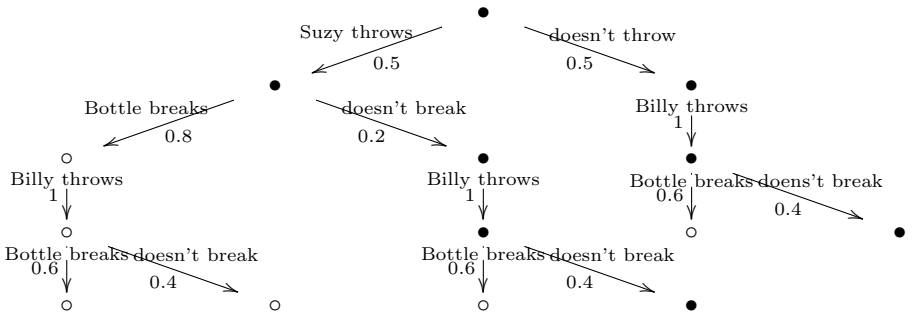
A theory in CP-logic [17] consists of a set of *CP-laws* of the form: $\forall \boldsymbol{x}\ (A_1 : \alpha_1) \vee \cdots \vee (A_n : \alpha_n) \leftarrow \phi$. Here, ϕ is a conjunction of literals and the A_i are atoms, such that the tuple of logic variables $\boldsymbol{x}$ contains all free logic variables in ϕ and the A_i. The α_i are non-zero probabilities with $\sum \alpha_i \leq 1$. Such a rule expresses that ϕ causes some (implicit) non-deterministic event, of which each A_i is a possible outcome with probability α_i. If $\sum_i \alpha_i = 1$, then at least one of the possible effects A_i must result if the event caused by ϕ happens; otherwise, the event may happen without any (visible) effect on the state of the world. For a CP-law r, we refer to ϕ as $\mathit{body}(r)$, and to the sequence $(A_i, \alpha_i)_{i=1}^n$ as $\mathit{head}(r)$.

The semantics of a theory in CP-logic is defined in terms of its grounding, so from now on we will restrict attention to ground theories, in which each tuple of logic variables $\boldsymbol{x}$ is empty. Any theory can be made ground by replacing the logic variables by constants. A ground atom can be considered as a binary random variable.

Example 2. Suzy and Billy may each decide to throw a rock at a bottle. Suzy throws with probability 0.5 and if she does, her rock breaks the bottle with probability 0.8. Billy always throws and his rock hits with probability 0.6.

$$(\mathit{Throws}(\mathit{Suzy}) : 0.5). \qquad (\mathit{Broken} : 0.8) \leftarrow \mathit{Throws}(\mathit{Suzy}).$$
$$(\mathit{Throws}(\mathit{Billy}) : 1). \qquad (\mathit{Broken} : 0.6) \leftarrow \mathit{Throws}(\mathit{Billy}).$$

The semantics of CP-logic is defined using the concept of an *execution model*. This is a probability tree in which each node s is labeled with a set of partial truth value assignments to atoms, which we denote as an *interpretation* $\mathcal{I}(s)$. Such trees are constructed, starting from a root node in which all atoms are false, by "firing" rules whose body holds. The following is an execution model for Example 2. States s in which the bottle is broken (i.e., $\mathcal{I}(s) \models Broken$) are represented by an empty circle, and those in which it is still whole by a full one.



Each such tree defines a probability distribution over its leaves, which induces a probability distribution over the interpretations $\mathcal{I}(s)$ that are associated to these leaves. A CP-theory may have many execution models, which differ in the order in which they fire rules. The differences between these trees are irrelevant, in the sense that they all produce the same probability distribution π_T in the end [17].

The above example can easily be represented as a Bayesian network, where $Broken$ is a noisy-or node with $Throws(Suzy)$ and $Throws(Billy)$ as its parents. This is in general the case for CP-theories that are acyclic [12]. Naively translating a CP-theory that is not acyclic to a Bayesian network would produce a cyclic graph.

The execution model semantics of CP-logic elegantly handles such cycles. As an example, we consider the following small instantiation of the previous example:

$$
\begin{aligned}
Inf(Alice) &\leftarrow InitialInf(Alice).\\
Inf(Bob) &\leftarrow InitialInf(Bob).\\
(Inf(Alice) : 0.2) &\leftarrow RiskyTravel(Alice).\\
(Inf(Bob) : 0.2) &\leftarrow RiskyTravel(Bob).\\
(Inf(Bob) : 0.6) &\leftarrow Inf(Alice).\\
(Inf(Alice) : 0.6) &\leftarrow Inf(Bob).
\end{aligned}
$$

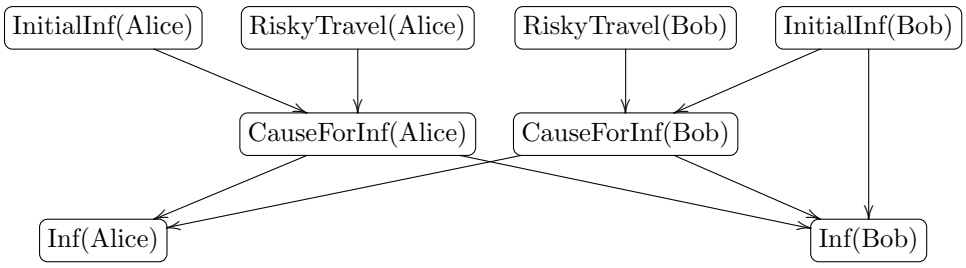
In the root of the execution model of this theory, $Inf(x)$ is still false for all x. It is only by applying the different rules that *Alice* and *Bob* may get infected. This ensure that the causal cycle between $Inf(Alice)$ and $Inf(Bob)$ is interpreted correctly and that, in particular, they cannot each cause the other to be infected

unless at least one of them was also initially infected or infected by risky travel. However, this same property also makes it tricky to interpret negation in rule bodies. For instance, suppose we also have a rule:

$$Quarantine(x) \leftarrow \neg Inf(x).$$

In the root of the tree, $\neg Inf(x)$ still holds for all x — including those for which $InitialInf(x)$ holds! Naive application of this rule could therefore lead us to conclude that also initially infected people need to be quarantined, which is clearly not intended. To solve this problem, each node s in an execution model not only keeps track of an interpretation $\mathcal{I}(s)$ that represents the actual state of the world in that node, but also of an overestimate $\mathcal{U}(s)$ that looks ahead in the causal process to see which atoms could potentially still be caused. As long as an atom $A \in \mathcal{U}(s)$, it is still possible that A will be caused further on in the tree, even if at the current node it is still the case that $\mathcal{I}(s) \not\models A$. While $A \in \mathcal{U}(s)$, a rule that depends on the negative literal $\neg A$ will therefore be prevented from firing. We omit the formal details of how this $\mathcal{U}(s)$ is computed, but they can be found in [17].

To translate cyclic CP-theories into Bayesian networks, it is typically necessary to introduce additional nodes:



In general, this translation requires the addition of n^2 of such new nodes in order to eliminate a cycle between n nodes. For large CP-theories, such a blow-up may render inference intractable. Moreover, because all of these new nodes are latent, they also make the network harder to interpret or learn. Finally, because this translation needs to consider the cycle as a whole, it may no longer be possible to update the resulting network in a modular way.

4 Bayesian Net Interpretation for Negation in the Head

We now investigate how the semantics of CP-logic can be extended to accomodate negative literals in the head. Before addressing this question in general, we first focus on a fragment of CP-logic that can be trivially translated into a Bayesian net, namely, that of ground, acyclic CP-theories in which each rule has only one atom in the head. We first show how the idea behind noisy-or can be extended to accomodate negative literals in this simple fragment, before investigating—in the next section—how this result can be extended to the whole of CP-logic.

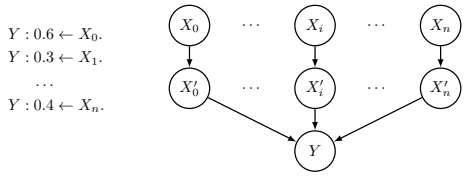


Fig. 1. CP-logic theory and the equivalent noisy-or Bayesian network

When adding a rule $(Y : \theta_i) \leftarrow X_i$ to a CP-logic theory, we increase the probability of Y being true given that condition X_i is true. This is equivalent to adding an additional parent X_i to a noisy-or construct (see Figure 1) and the probability of Y given the parents X_i can be calculated with:

$$Pr(Y = \top \mid X_{0:n}) = 1 - \prod_{\substack{i \in [0,n] \\ X_i = \top}} (1 - \theta_i) = \sum_{\substack{i \in [0,n] \\ X_i = \top}} \overbrace{\prod_{\substack{j \in [0,i[\\ X_j = \top}} (1 - \theta_j)}^{\text{weighing}} \cdot \theta_i$$

To adhere to the laws of probability, the total probability that Y is true should be equal or less than 1.0. The noisy-or structure achieves this by *weighing* each contribution of a parent by the remainder of the total probability of the parents already taken into account. The weighing expresses the probability that the variable is not true due to any of the previous probabilities (see also Figure 2).

Suppose we now add a rule $(\neg Y : \theta_{n+1}) \leftarrow X_{n+1}$, which expresses a reduction of the probability that Y is true if X_i is true. In this case we need to ensure that the probability of Y is equal or larger than 0. Similar to noisy-or, we can achieve this by weighing the probability we are subtracting. In this case, the weighing factor is the total probability that Y is true because of any of the previous (positive) parents (see Figure 2).

$$\begin{aligned} Pr(Y = \top \mid X_{0:n}, X_{n+1} = \top) &= Pr(Y = \top \mid X_{0:n}) - \overbrace{Pr(Y = \top \mid X_{0:n})}^{\text{weighing}} \cdot \theta_{n+1} \\ &= Pr(Y = \top \mid X_{0:n}) \cdot (1 - \theta_{n+1}) \end{aligned}$$

When adding multiple rules with negative literals in the head $\neg Y : \theta_{n+1} \leftarrow X_{n+1}., \ldots, \neg Y : \theta_m \leftarrow X_m$, the computation of the probability of Y can be generalized to:

$$Pr(Y = \top \mid X_{0:m}) = Pr(Y = \top \mid X_{0:n}) \cdot \left(1 - \sum_{\substack{i \in]n,m] \\ X_i = \top}} \prod_{\substack{j \in]n,i[\\ X_j = \top}} (1 - \theta_j) \cdot \theta_i \right) \quad (3)$$

$$= Pr(Y = \top \mid X_{0:n}) \cdot (1 - Pr(Y = \bot \mid X_{n+1:m})) \quad (4)$$

To represent Formula 4 as a Bayesian net it must to be expressed as a set of conditional probability tables. Given two auxiliary variables P and N with

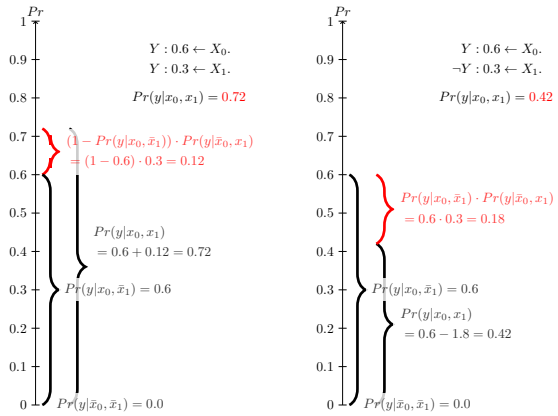


Fig. 2. Interpretation for probability scaling. For brevity we write $X = \top$ as x and $X = \bot$ as $\bar{x}$.

respectively the noisy-or structures $Pr(Y \mid X_{0:n})$ and $Pr(Y \mid X_{n+1:m})$, the formula $Pr(Y \mid X_{0:m})$ can now be written as $Pr(Y \mid P, N)$. The conditional probability table for this last conditional probability distribution is equivalent to $Y \Leftrightarrow P \wedge \neg N$ (see Figure 3).

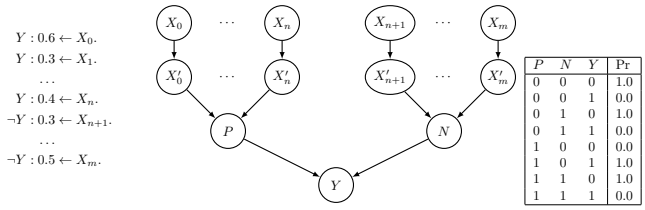


Fig. 3. Bayesian network that efficiently encodes an inhibited noisy-or. P is a noisy-or for $X_0, \ldots, X_n$ and N for $X_{n+1}, \ldots, X_m$.

5 Generalization to CP-logic Programs

We now examine how we can incorporate the intuitions of the previous section into the general setting of CP-logic. To be more precise, from now on, we allow rules of the form:

$$\forall \boldsymbol{x} \quad (L_1 : \alpha_1) \vee \cdots \vee (L_n : \alpha_n) \leftarrow \phi.$$

Here, ϕ is again a first-order logic formula with $\boldsymbol{x}$ as free logic variables and the $\alpha_i \in [0, 1]$ are again such that $\Sigma \alpha_i \leq 1$. Each of the L_i is now either a *positive effect literal* A (i.e., an atom) or a *negative effect literal* $\neg A$.

While the goal of this extension is of course to be able to represent such phenomena as described in Section 2, let us first take a step back and consider, in the abstract, which possible meanings this construct could reasonably have. Clearly, if for some atom A only positive effect literals are caused, the atom should end up being true, just as it always has. Similarly, if only negative effect literals $\neg A$ are caused, the atom A should be false. However, this does not even depend on the negative effect literals being present: because false is the default value in CP-logic, an atom will already be false whenever there are no positive effect literals for it, even if there are no negative effect literals either.

The only question, therefore, is what should happen if, for some A, both a positive and a negative effect literal are caused. One alternative could be that the result would somehow depend on the relative strength of the negative and positive effects, e.g., whether the power of aspirin to prevent a fever is "stronger" than the power of flu to cause it. However, such a semantics would be a considerable departure from the original version of CP-logic, in which cumulative effects (synergy, interference, ...) are strictly ignored. In other words, CP-logic currently makes no distinction whatsoever between a headache that is simultaneously caused by five different conditions and a headache that has just a single cause. This design decision was made to avoid a logic that, in addition to probabilities, would also need to keep track of the degree to which a property holds. A logic combining probabilities with such fuzzy truth degrees would, in our opinion, become quite complex and hard to understand.

In this paper, we want to preserve the relative simplicity of CP-logic, and we will therefore again choose not to work with degrees of truth. Therefore, only two options remain: when both effect literals A and $\neg A$ are caused, the end result must be that A is either true of false. This basically means that, in the presence of both kinds of effect literals, we have to ignore one kind. It is obvious what this choice should be: the negative effect literals already have no impact on the semantics when there are only positive effect literals or when there are no positive effect literals, so if they would also have no impact when positive and negative effect literals are both present, then they would have never have any impact at all and we would have introduced a completely superfluous language construct. Therefore, the only reasonable choice is to give negative effect literals precedence over positive ones, that is, an atom A will be true if and only if it is caused at least once and no negative effect literal $\neg A$ is caused.

This can be formally defined by a minor change to the existing semantics of CP-logic. Recall that, in the current semantics, each node s of an execution model has an associated interpretation $\mathcal{I}(s)$, representing the current state of the world, and an associated three-valued interpretation $\mathcal{U}(s)$, representing an overestimate of all that could still be caused in s. We now add to this a third set, namely a set of atoms $\mathcal{N}(s)$, containing all atoms for which a negative effect literal has already been caused. The sets $\mathcal{I}(s)$ and $\mathcal{N}(s)$ evolve throughout an execution model as follows:

- In the root of the tree, $\mathcal{I}(s) = \mathcal{N}(s) = \{\}$

– When a *negative* effect literal $\neg A$ is caused in a node s, the execution model adds a child s' to s such that:
  - $\mathcal{N}(s') = \mathcal{N}(s) \cup \{A\}$;
  - $\mathcal{I}(s') = \mathcal{I}(s) \setminus \{A\}$.
– When a *positive* effect literal A is caused in a node s, the execution model adds a child s' to s such that:
  - $\mathcal{N}(s') = \mathcal{N}(s)$;
  - if $A \in \mathcal{N}(s)$, then $\mathcal{I}(s') = \mathcal{I}(s)$, else $\mathcal{I}(s') = \mathcal{I}(s) \cup \{A\}$.

Note that, throughout the execution model, we maintain the property that $\mathcal{N}(s) \cap \mathcal{I}(s) = \{\}$.

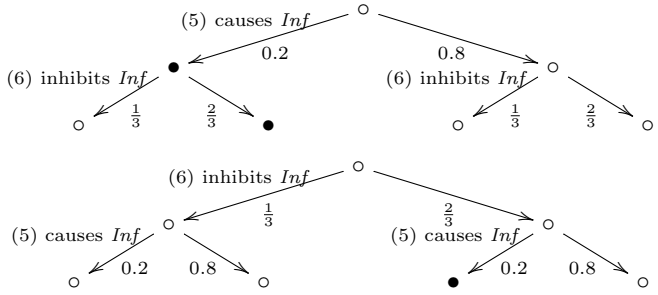
The overestimate $\mathcal{U}(s)$ is still constructed in the usual way (see [17]), with the exception that atoms from $\mathcal{N}(s)$ may no longer be added to it.

To illustrate, let us consider the following simple example, where we assume that *Alice* belongs to both exogenous predicates *RiskyTravel* and *Resistant*:

$$(\mathit{Inf}(\mathit{Alice}) : 0.2) \leftarrow \mathit{RiskyTravel}(\mathit{Alice}). \tag{5}$$

$$(\neg\mathit{Inf}(\mathit{Alice}) : \frac{1}{3}) \leftarrow \mathit{Resistant}(\mathit{Alice}). \tag{6}$$

Representing nodes in which *Alice* is infected by a full circle, these two rules may produce either of the following two execution models.



Again, the differences between these two execution models are irrelevant, because they both produce a distribution over final states in which $P(\mathit{Inf}(\mathit{Alice})) = \frac{2}{3}{\cdot}0.2$, which is of course the same probability as we obtain with formula 4 of the previous section. Note that, in order to obtain this property, it is important that inhibition always trumps causation, regardless of which happens first. Unlike formula 4, the execution model semantics is equally applicable to cases with cyclic causation.

To implement this feature of negation-in-the-head, a simple transformation to regular CP-logic may be used. This transformation is based on the way in which [3] encode causal ramifications in their inductive definition modelling of the situation calculus.

For a CP-theory T in vocabulary Σ, let $\Sigma_\neg$ consist of all atoms A for which a negative effect literal $\neg A$ appears in T. For each atom $A \in \Sigma_\neg$, we introduce two new atoms, C_A and $C_{\neg A}$. Intuitively, C_A means that there is a cause for A, and $C_{\neg A}$ means that there is a cause for $\neg A$. Let τ_A be the following transformation:

- Replace all positive effect literals A in the heads of rules by C_A
- Replace all negative effect literals $\neg A$ in the heads of rules by $C_{\neg A}$
- Add this rule: $A \leftarrow C_A \wedge \neg C_{\neg A}$

Let $\tau_\neg(T)$ denote the result of applying to T, in any order, all the transformations τ_A for which $A \in \Sigma_\neg$. It is clear that $\tau_\neg(T)$ is a regular CP-theory, i.e., one without negation-in-the-head. As the following theorem shows, this reduction preserves the semantics of the theory.

Theorem 1. *For each interpretation X for the exogenous predicates, the projection of $\pi^X_{\tau_\neg(T)}$ onto the original vocabulary Σ of T is equal to π^X_T.*

When comparing the transformed theory $\pi_{\tau_\neg}(T)$ to the original theory T, we see that the main benefit of having negation-in-the-head lies in its *elaboration tolerance*: there is no need to know before-hand for which atoms we later might wish to add negative effect literals, since we can always add these later, without having to change to original rules.

6 Application: Encoding Interventions

One of the interesting uses of negation-in-the-head is related to the concept of interventions, introduced by [13]. Let us briefly recall this notion. Pearl works in the context of *structural models*. Such a model is built from a number of random variables. For simplicity, we only consider Boolean random variables, i.e., atoms. These are again divided into exogenous and endogenous atoms. A structural model now consists of one equation $X := \varphi$ for each endogenous atom X, which defines that X is true if and only if the boolean formula φ holds. This set of equations should be acyclic, in order to ensure that an assignment of values to the exogenous atoms induces a unique assignment of values to the endogenous ones.

A crucial property of causal models is that they can not only be used to predicts the normal behaviour of a system, but also to predict what would happen if outside factors unexpectedly intervene with its normal operation. For instance, consider the following simple model of which students must repeat a class:

$$\mathit{Fail} := \neg \mathit{Smart} \wedge \neg \mathit{Effort}. \qquad \mathit{Repeat} := \mathit{Fail} \wedge \mathit{Required}.$$

Under the normal operation of this "system", only students who are not smart can fail classes and be forced to repeat them. Suppose now that we catch a student cheating on an assignment and decide to fail him for the class. This action was not foreseen by the causal model, so it does not follow from the normal behaviour. In particular, failing the student may cause him to have to repeat the class, but if the student is actually smart, then failing him will not make him stupid. Pearl shows that we can model our action of failing the student by means of an *intervention*, denoted $do(\mathit{Fail} = \top)$. This is a simple syntactic transformation, which removes and replaces the original equation for Fail:

$$\mathit{Fail} := \top. \qquad \mathit{Repeat} := \mathit{Fail} \wedge \mathit{Required}.$$

According to this updated set of equations, the student fails and may have to repeat the class, but he has not been made less smart.

In the context of CP-logic, let us consider the following simple medical theory:

$$(\mathit{HighBloodPressure} : 0.6) \leftarrow \mathit{BadLifeStyle}. \tag{7}$$

$$(\mathit{HighBloodPressure} : 0.9) \leftarrow \mathit{Genetics}. \tag{8}$$

$$(\mathit{Fatigue} : 0.3) \leftarrow \mathit{HighBloodPressure}. \tag{9}$$

Here, *BadLifeStyle* and *Genetics* are two exogenous predicates, which are both possible causes for *HighBloodPressure*. Suppose now that we observe a patient who suffers from *Fatigue*. Given our limited theory, this patient must be suffering from *HighBloodPressure*, caused by at least one of its two possible causes.

Now, suppose that a doctor is wondering whether it is a good idea to prescribe this patient some pills that lowers high blood pressure. Again, the proper way to answer such a question is by means of an *intervention*, that first prevents the causal mechanisms that normally determine someone's blood pressure and then substitutes a new "mechanism" that just makes *HighBloodPressure* false. This can be achieved by simply removing the two rules (7) and (8) from the theory. This is an instance of a general method, developed in [16], of performing Pearl-style interventions in CP-logic. The result is that probability of *Fatigue* drops to zero, i.e., $P(\mathit{Fatigue} \mid do(\neg \mathit{HighBloodPressure})) = 0$.

In this way, we can evaluate the effect of prescribing the pills *without* actually having these pills in our model. This is a substantial difference to the way in which reasoning about actions is typically done in the field of knowledge representation, where formalisms such as situation or event calculus require an explicit enumeration of all available actions and their effects. Using an intervention, by contrast, we can envisage the effects of actions that we never even considered when writing our model.

Eventually, however, we may want to transform the above *descriptive* theory into a *prescriptive* one that tells doctors how to best treat a patient, given his or her symptoms. In this case, we would need rules such as this:

$$\mathit{BPMedicine} \leftarrow \mathit{Fatigue}. \tag{10}$$

Obviously, this requires us to introduce the action *BPMedicine* of prescribing the medicine, which previously was implicit in our intervention, as an explicit action in our vocabulary. Negation-in-the-head allows us to syntactically express the effect of this new action: $\neg \mathit{HighBloodPressure} \leftarrow \mathit{BPMedicine}$.

This transformation can be applied in general, as the following theorem shows.

Theorem 2. *Let T be a CP-theory over a propositional vocabulary Σ. For an atom $A \in \Sigma$, let T' be the theory $T \cup \{r\}$ with r the rule $\neg A \leftarrow B$ and B an exogenous atom not in Σ. For each interpretation X for the exogenous atoms of T', if $B \in X$, then $\pi^X_{T'} = \pi^X_{do(T,\neg A)}$ and if $B \notin X$, then $\pi^X_{T'} = \pi^X_T$.*

This theorem shows that negation-in-the-head allows CP-theories to "internalize" the intervention of *doing* $\neg A$. The result is a theory T' in which the

intervention can be switched on or off by simply choosing the appropriate interpretation for the exogenous predicate that now explicitly represents this intervention. Once the intervention has been syntactically added to the theory in this way, additional rules such as (10) may of course be added to turn it from an exogenous to an endogenous property.

It is important to note that this is a fully modular and elaboration tolerant encoding of the intervention, i.e., the original CP-theory is left untouched and the rules that describe the effect of the intervention-turned-action are simply added to it. This is something that we can only achieve using negation-in-the-head.

7 Experiments

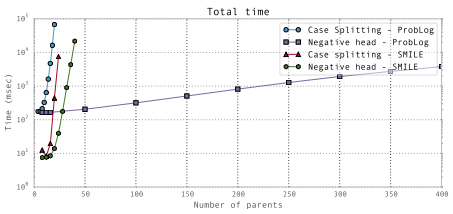
We have presented an intuitive and modular approach to express an inhibition structure. In this section, we evaluate the computational cost associated with this alternative structure. For this we perform inference on three theories: (i) the inhibited noisy-or structure from Fig. 3, (ii) the inhibited noisy-or structure translated to case splitting, and (iii) the infection example from Section 2. Inference was performed using ProbLog[1], an SRL system to which CP-logic can be compiled, for all three theories and using SMILE[2], a state-of-the-art PGM toolbox for the first two acyclic theories. All experiments are run on a 3GHz Intel Core2 Duo CPU with 2GB memory and timings are averaged over 3 runs.

For the inhibited noisy-or structure, the inference can be linear depending on the encoding of the noisy-or substructures [15, 5]. The results (fig. 4a) show that the use of negative effect literals, implemented by means of their noisy-or encoding, is always more efficient than case splitting. Surprisingly, when using SMILE the inference has exponential complexity with a growing number of parents. This indicates that, although we used noisy-max encodings, noisy-or is not fully exploited. ProbLog is able to exploit the local structure more efficiently and performs inference for the inhibited noisy-or in time polynomial in the number of parents.

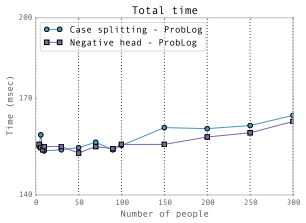
The infection example contains cycles and can therefore only be processed by ProbLog. For this theory, we let the number of people increase, while keeping the number of contacts per person fixed (Fig. 4b). This increases the number of inhibited noisy-or structures but, contrary to the previous theory, not the number of parents. We see that the version using case splitting is slower with approximately a constant factor.

We can conclude that the overhead introduced by the encoding for negative literals in the head is marginal compared to normal noisy-or combinations and inference can be performed efficiently.

[1] http://dtai.cs.kuleuven.be/problog
[2] http://genie.sis.pitt.edu



(a) Inference for an inhibited noisy-or structure.



(b) Inference for the infection example.

Fig. 4. Runtime of inference.

8 Related Work

8.1 Inhibited Recursive Noisy-or

The structure we obtain is related to the inhibited recursive noisy-or structure [10] which states that:

$$Pr(Y|\mathbf{X}) = Pr(Y \text{ caused by } \mathbf{X}) \cdot Pr(Y \text{ not inhibited by } \mathbf{X})$$

The two parts are recursive noisy-or models, a generalisation of noisy-or that relaxes the ICI assumption. It allows to encode synergies, the combination of two causes to have a stronger effect than expected, and interferences, the combination to have a softer effect. A problem, however, with recursive noisy-or models is that the parametrisation may be asymmetric. As this causes confusion and conflicts, this model does not allow for a modular representation and is not popular in common use [4]. Different in CP-logic is that concepts like synergy and interference are not represented using a recursive parametric probability distribution but directly in the program using the conditions in the body and positive and negative literals. As such, CP-logic, offers a modular and non-conflicting alternative to inhibited recursive noisy-or models.

8.2 The Certainty Factor Model

Rule-based systems are popular for expert and diagnostics systems because they offer an intuitive syntax to human experts. In this setting, the concept of weighing the level of uncertainty of inhibiting factors has been proposed for *certainty factors* used in the MYCIN system [1, 11]. The weighing, however, is performed independently for the measures of belief and disbelief and are joined only afterwards to define the certainty factor. These notions of uncertainty are not well-founded from a probabilistic point of view but are used in practice because they are computationally simple and behave satisfactorily. It was argued that the Bayesian framework was unsatisfactory because it would require too many conditional probability parameters that have to be filled in by an expert. This was a motivation to use the two different measures, one for belief and one

for disbelief. The simplicity of the certainty factor model, however, was achieved only with frequently unrealistic assumptions and with persistent confusion about the meaning of the numbers being used [8]. Heckerman and Shortlife show how Bayesian nets can be used to represent the certainty factor model in a principled manner. Unfortunately, they show that "uncertain reasoning is inherently less modular than is logical reasoning", which is an attractive feature of the certainty factor model. In this work we show that both concepts of belief and disbelief can be represented in one rule-based framework with a strong foundation in probability theory and with the modularity properties of logical reasoning.

8.3 Interaction Rules in Probabilistic Logic

Negation in the head can be interpreted as a modification of the noisy-or interaction rule that is common among probabilistic logics. *Probabilistic interaction logic* [9] is a framework that generalizes languages like CP-logic and ProbLog to allow custom encodings of the interaction rules. This is achieved by building on top of default logic instead of logic programming. Part of the example in Section 2 can be expressed as:

$$D = \{\frac{RiskyTravel(x) \wedge p(x) \ : \ Inf(x)}{Inf(x)}\}, \quad W = \{Resistant(x) \wedge q(x) \rightarrow \neg Inf(x)\}$$

with $P(p(x)) = 0.2$ and $P(q(x)) = 1/3$. Here, the single default in D expresses that, if x has done risky travel, this will cause her to be infected with probability 0.2, *unless we know otherwise*. The implication in W then gives precisely such a reason for knowing otherwise, namely, the fact that x might be resistant.

This logic is obviously quite general, allowing many more interaction patterns to be expressed than just the simple inhibited effects we have considered here. However, it does depend on the user to correctly encode these patterns in first-order logic: for instance, adding the inhibiting effect of being resistant will require a change to the original theory, unless the user had the foresight to already include the justification $Inf(x)$ in his original default.

9 Conclusion

In this paper, we have presented the new language feature of negative effect literals. We have shown this for the case of CP-logic where it offers a natural extension the capacity to represent causal models in a modular way. In the particular case of theories that correspond to a Bayesian net, such negative effect literals correspond to an inhibited noisy-or structure. Additionally, we show that this new language feature can be encoded in such a manner that inference can be performed with a complexity similar to standard noisy-or.

References

1. Buchanan, B.G., Shortlife, E.H.: Rule Based Expert Systems: The MYCIN Experimehts of the Stanford Heuristic Programming Project. Addison-Wesley (1984)

2. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic Inductive Logic Programming - Theory and Applications, vol. 4911. Springer, Heidelberg (2008)
3. Denecker, M., Ternovska, E.: Inductive situation calculus. Artificial Intelligence 171(5-6), 332–360 (2007)
4. Díez, F.J., Druzdzel, M.: Canonical probabilistic models for knowledge engineering. Technical report cisiad-06-01, UNED, Madrid, Spain (2006)
5. Díez, F.J., Galán, S.F.: Efficient computation for the noisy max. International Journal of Intelligent Systems 18(2), 165–177 (2003)
6. Fierens, D., Van den Broeck, G., Thon, I., Gutmann, B., De Raedt, L.: Inference in probabilistic logic programs using weighted CNFs. arXiv preprint arXiv:1202.3719 (2012)
7. Getoor, L., Taskar, B. (eds.): Statistical Relational Learning. MIT Press (2007)
8. Heckerman, D.E., Shortliffe, E.H.: From certainty factors to belief networks. Artificial Intelligence in Medicine 4(1), 35–52 (1992)
9. Hommersom, A., Lucas, P.J.F.: Generalising the interaction rules in probabilistic logic. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two. pp. 912–917 (2011)
10. Kuter, U., Nau, D., Gossink, D., Lemmer, J.F.: Interactive course-of-action planning using causal models. In: Third International Conference on Knowledge Systems for Coalition Operations (KSCO-2004) (2004)
11. Lucas, P., Van Der Gaag, L.: Principles of expert systems. Addison-Wesley Longman Publishing Co., Inc. (1991)
12. Meert, W., Struyf, J., Blockeel, H.: Learning ground CP-logic theories by leveraging Bayesian network learning techniques. Fundamenta Informaticae 89(1), 131–160 (2008)
13. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press (2000)
14. Sato, T., Kameya, Y.: New advances in logic-based probabilistic modeling by PRISM. In: De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.) Probabilistic Inductive Logic Programming. Lecture Notes in Computer Science, vol. 4911, pp. 118–155. Springer, Heidelberg (2008)
15. Takikawa, M., D'Ambrosio, B.: Multiplicative factorization of noisy-max. In: Laskey, K.B., Prade, H. (eds.) Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI). pp. 622–630 (1999)
16. Vennekens, J., Bruynooghe, M., Denecker, M.: Embracing events in causal modelling: Interventions and counterfactuals in CP-logic. In: Janhunen, T., Niemel, I. (eds.) Logics in Artificial Intelligence. Lecture Notes in Computer Science, vol. 6341, pp. 313–325. Springer, Heidelberg (2010)
17. Vennekens, J., Denecker, M., Bruynooghe, M.: CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming (TPLP) 9(3), 245–308 (2009)
18. Zagorecki, A., Druzdzel, M.J.: An empirical study of probability elicitation under noisy-or assumption. In: FLAIRS Conference. pp. 880–886 (2004)